

# *eunix: echo*

Eric Bailey

October 31, 2017 <sup>1</sup>

<sup>1</sup> Last updated October 31, 2017

A reimplementaion of echo for my own edification.

## *Headers and Forward Declarations*

Include the core input and output functions from the C standard library.

1a `<* 1a>≡  
#include <stdio.h>`

This definition is continued in chunks 1 and 3e.  
Root chunk (not used in this document).

Describe GNU getopt

1b `<* 1a>+≡  
#include <getopt.h>`

Declare the `usage` function.

1c `<* 1a>+≡  
void usage();`

Uses `usage` 3e.

## *The main Function*

1d `<* 1a>+≡  
int main(int argc, char *argv[])  
{  
 <Process given options. 2d>  
  
 <Print each string, separated by a space. 3d>  
  
 <Print a newline unless the -n option was given. 2b>  
  
 return 0;  
}`

Defines:  
`main`, never used.

## Processing Options

Currently, the *⟨legal options 2a⟩* are:

- `-n` (do not print a trailing newline)

2a *⟨legal options 2a⟩*≡  
`n`

This code is used in chunk 3a.

*-n (do not print a trailing newline)*

Declare a variable `newline_flag` to determine whether or not to print a newline after printing the rest of the given strings, i.e.

2b *⟨Print a newline unless the -n option was given. 2b⟩*≡  
`if (newline_flag)  
 putchar('\n');`

This code is used in chunk 1d.

Uses `newline_flag` 2d.

When the `-n` option is given, set `newline_flag` to 0, thereby disabling the printing of the trailing newline.

2c *⟨Handle -n. 2c⟩*≡  
`case 'n':  
 newline_flag = 0;  
 break;`

This code is used in chunk 2e.

Uses `newline_flag` 2d.

By default, print a trailing newline.

2d *⟨Process given options. 2d⟩*≡  
`/* Flag set by '-n'. */  
int newline_flag = 1;`

This definition is continued in chunk 2e.

This code is used in chunk 1d.

Defines:

`newline_flag`, used in chunk 2.

## Looping Through Given Options

2e *⟨Process given options. 2d⟩*+≡  
`int c;  
  
while ((⟨Process each option until EOF. 3a⟩) {  
 switch (c) {  
 ⟨Handle -n. 2c⟩  
 ⟨Handle illegal options. 3b⟩  
 }  
}`

This code is used in chunk 1d.

Defines:

`c`, used in chunk 3a.

Describe this, esp. getopt

3a *<Process each option until EOF. 3a>*≡  
 (c = getopt(argc, argv, "*<legal options 2a>*")) != EOF

This code is used in chunk 2e.  
 Uses c 2e.

If the user gives an illegal option, i.e. one not included in the *<legal options 2a>*, display the **usage** information and return a non-zero status code.

3b *<Handle illegal options. 3b>*≡  
 case '?':  
     usage();  
     return 1;

This code is used in chunk 2e.  
 Uses usage 3e.

## Echoing Strings

Otherwise, loop through the remainder of argv and print each string, followed by a space. Unless the current string is the last one, i.e. index == argc - 1, in which case, do **not** print a space.

3c *<Write a space unless this is the last string. 3c>*≡  
 if (index < argc - 1)  
     putchar(' ');

This code is used in chunk 3d.  
 Uses index 3d.

Describe optind

3d *<Print each string, separated by a space. 3d>*≡  
 int index;  
  
 for (index = optind; index < argc; index++) {  
     printf("%s", argv[index]);  
     *<Write a space unless this is the last string. 3c>*  
 }

This code is used in chunk 1d.  
 Defines:  
 index, used in chunk 3c.




## The usage Function

Display information on how to use echo, including *<legal options 2a>*.

3e *<\* 1a>*+≡  
 void usage()  
 {  
     printf("Usage: echo [-n] [string ...]\n");  
 }

Defines:  
 usage, used in chunks 1c and 3b.

*To-Do*

 Describe GNU <code>getopt</code> . . . . .	1
 Describe this, esp. <code>getopt</code> . . . . .	3
 Describe <code>optind</code> . . . . .	3